Jeu du Billard en JavaScript

Huang Véronique
Moreira-Faria Dylan
Bensiam Simon

Pour ce projet en Web Avancée, nous avons réalisé en groupe de trois le jeu du billard Anglais.
Les règles du billard Anglais sont :
-Si la première boule touché n'est pas une boule de ma couleur, le tour suivant l'adversaire peut replacer la boule où il veut sur le plateau et jouer par la suite.
-Le premier à faire tomber toutes ces boules de couleur et ensuite la boule noire à gagner la partie.
-Si un adversaire fait tomber la boule noire avant toutes ces boules de couleurs, sont adversaire gagne.

Area.js

Pour implémenter ce jeu, nous nous sommes d'abord occupé du jeu. Après plusieurs versions, nous avons créer une classe Area qui permet de redimensionner tous les objets présent dans le jeu dont elle possède une méthode resize() et une méthode draw() pour afficher des images.

classes.js / index.js

Dans le fichier classes.js du côté client, nous avons trois classes différentes.
La classe Ball permet de créer une boule avec une origine (x et y), une variable out pour savoir si la boule se trouve encore sur le plateau, une couleur, une variable ismoving si la boule est en train de bouger, un vx et vy pour la vélocité, un rayon, une masse, une image et une area utilisé dans la méthode draw() qui permet la redimension de l'image.

La classe Stick est la classe pour la queue, comme la boule, elle possède une origine (x et y), une image, mais aussi une rotation et une autre origin pour la rotation de l'image et d'une variable out si la queue n'est pas présente dans le plateau. Dans cette classe la méthode draw() permet d'afficher la queue mais aussi de tourner l'image par rapport à un angle donné.

Pour la classe Player, c'est une une classe qui permet l'affichage du score des joueurs, le name est le nom du joueur, area est une variable pour la redimension, une couleur, une origine pour l'icon du joueur et un nombre ballOut pour le nombre de boule empoché par le joueur.

main.js

Le fichier main du côté du client gère la plupart de son propre affichage. Lorsqu'un utilisateur se connecte, il se retrouve dans un lobby dont il peut attendre d'autre joueur. Lorsqu'un autre joueur se connecte, le premier joueur peut venir jouer avec lui en appuyant sur le bouton "défier", le client envoie ensuite un message au serveur. Lorsque le serveur reçoit ce message il envoie la fonction onStart() (index.js) qui va envoyer les deux utilisateurs dans une même partie dans la boucle gameloop.

La boucle gameloop contient toute les règles du jeu, le premier joueur commence la partie, après le chargement de la page, il peut alors cliquer et glisser la souris dans la direction qu'il veut que la boule blanche aille, pour implémenter cela, il se trouve dans le fichier main.js. Dans la fonction playerMove(mouseX,MouseY) cette fonction gère le calcul de la rotation de la queue par rapport à chaque coordonnée de la souris. Pour envoyer la boule, l'utilisateur doit double cliquer dans la direction voulu, c'est la fonction playerClick(mouseX,mouseY) qui s'occupe d'envoyer la boule dans la direction donnée, ladirection de la rotation de la boule est calculé par la variable angle et le power par la queue, ces deux données sont envoyé au serveur qui va ensuite faire appel à la fonction _onPlayerAction() qui va faire tous les calculs de la vitesse de la boule et de sa direction à prendre.

Quand au déplacement des boules, la vélocité est calculé dans le serveur et chaque coordonnés de chaques balles est envoyé au client par le message "state", le client n'y a plus cas afficher les boules aux bonnes coordonnées.

Pour les collisions, si cette évènement se produit avec deux ou plusieurs boules, c'est dans la boucle principale, gameloop, collideWith() qui va pouvoir vérifier pour chaque boule si elles sont entré en collision, si oui, alors la fonction collideWith() va calculer la vélocité de la boule et de la boule cogné.

Pour vérifier si une boule est en collision avec une autre, nous faisons le calcul de leurs distance entre elle, si cette distance est inférieur à leurs somme de leur rayon, alors elles sont en collision, ainsi nous obtenons leurs nouvelles vélocité à chacunes.
Cette condition suivante vérifie si une des boules rentré dans une poche (trou).

```
for(var i=0; i<holes.length; i++){
    …
    if( distance_ < (this.radius+holes[i].radius)){...}
}
```

Nous mettons ensuite cette boule empoché en dehors du plateau (out = true) et nous la mettons en coordonnée (0;0) pour ne pas affecter d'autre collision avec celle la, cette boule n'est alors plus affiché dans le jeu et la boule appartenant au joueur est ajouté à son score. Le tour du joueur est fini lorsque toutes boules sont à l'arrêt et le second joueur peut alors maintenant jouer.

Dans les règles du billard anglais, lorsqu'un utilisateur fait tomber la boule blanche dans une poche, c'est au tour de son adversaire de joueur et de replacer la balle là où il le désire. C'est alors dans la fonction _onPlayerAction() que se trouve l'implémentation de cette règle. La première condition `if(!this.boards[roomId][15].out){...}` vérifie si la boule blanche se trouve sur le plateau (variable out), si la variable out est vrai,

```
if(this._checkBall(x, y, this.boards[roomId][15], this.boards[roomId],
this.holes)==true){ ...}
```

alors le joueur va devoir double cliquer dans un endroit, si cette endroit est en collision avec une autre boule ou qu'elle se trouve en dehors du plateau, le joueur va devoir trouver un autre emplacement.

En conclusion, le jeu est implémenté comme il le fallait, mais il reste encore quelques problème, comme celle des collisions, dont certaines boules puissent se chevaucher en plein calculs. De plus la queue s'affiche aussi sur les deux plateaux des joueurs à chaque fois et nous n'avions pas réussi à afficher une boîte de dialogue lorsque un des joueurs à gagner. Les envoies des données sont correct aussi bien dans le lobby que dans le jeu. Implémenter ce jeu à été intéressant mais ce projet nous a aussi permit d'en apprendre d'avantage sur le langage JavaScript et sur le réseaux.